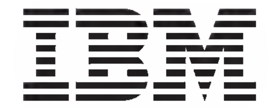
IBM Tivoli Directory Server

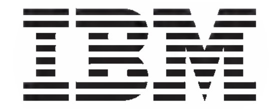


Performance Tuning Guide

Version 5.2

SC32-1342-00

IBM Tivoli Directory Server



Performance Tuning Guide

Version 5.2

SC32-1342-00



First Edition (October, 2003)

This edition applies to version 5, release 2, of IBM Tivoli Directory Server and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Preface

Welcome to the *IBM® Tivoli® Directory Server Version 5.2 Performance Tuning Guide*. The purpose of this document is to provide information about improving performance for IBM Tivoli Directory Server.

For the most current and accurate tuning information, see the Web version of the *IBM Tivoli Directory Server Version 5.2 Performance Tuning Guide*.

See "Accessing publications online" on page vi for information about accessing online publications.

Who should read this book

The target audience for this guide includes:

- System installation and deployment administrators
- Network system administrators
- Information Technology architects
- Application developers

Publications

Read the descriptions of the IBM Tivoli Directory Server library to determine which publications you might find helpful. After you determine the publications you need, see "Accessing publications online" on page vi.

IBM Tivoli Directory Server library

The publications in the IBM Tivoli Directory Server library are:

IBM Tivoli Directory Server Version 5.2 Readme Addendum
: Go to the Tivoli Software Library Web site to access the *IBM Tivoli Directory Server Version 5.2 Readme Addendum*, which contains important information that was not included in the Readme files. See "Accessing publications online" on page vi for information about accessing online publications.

IBM Tivoli Directory Server Version 5.2 Client Readme
: Contains last-minute information about the client.

IBM Tivoli Directory Server Version 5.2 Server Readme
: Contains last-minute information about the server.

IBM Tivoli Directory Server Version 5.2 Web Administration Tool Readme
: Contains last-minute information about the Web Administration Tool. This Readme is available from the main panel of the Web Administration Tool.

IBM Tivoli Directory Server Version 5.2 Installation and Configuration Guide
: Contains complete information for installing the IBM Tivoli Directory Server client, server, and Web Administration Tool. Includes information about migrating from a previous version of IBM Tivoli Directory Server or SecureWay® Directory.

IBM Tivoli Directory Server Version 5.2 Tuning Guide
: Contains information about tuning the server for better performance.

IBM Tivoli Directory Server Version 5.2 Administration Guide
Contains instructions for performing administrator tasks through the Web Administration Tool or the command line.

IBM Tivoli Directory Server Version 5.2 Plug-ins Reference
Contains information about writing server plug-ins.

IBM Tivoli Directory Server Version 5.2 C-Client SDK Programming Reference
Contains information about writing LDAP client applications.

Related publications

Information related to the IBM Tivoli Directory Server is available in the following publications:

- IBM Tivoli Directory Server Version 5.2 uses the Java™ Naming and Directory Interface (JNDI) client from Sun Microsystems. For information about the JNDI client, refer to the *Java Naming and Directory Interface™ 1.2.1 Specification* on the Sun Microsystems Web site at http://java.sun.com/products/jndi/1.2/javadoc/index.html.
- The Tivoli Software Library provides a variety of Tivoli publications such as white papers, datasheets, demonstrations, redbooks, and announcement letters. The Tivoli Software Library is available on the Web at: http://www.ibm.com/software/tivoli/library/
- The *Tivoli Software Glossary* includes definitions for many of the technical terms related to Tivoli software. The *Tivoli Software Glossary* is available, in English only, from the **Glossary** link on the left side of the Tivoli Software Library Web page http://www.ibm.com/software/tivoli/library/
- The IBM DB2 Universal Database™ documentation can be found at the following Web site: http://www.ibm.com/software/data/db2/library/

Accessing publications online

The publications for this product are available online in Portable Document Format (PDF) or Hypertext Markup Language (HTML) format, or both in the Tivoli software library: http://www.ibm.com/software/tivoli/library

To locate product publications in the library, click the **Product manual**s link on the left side of the library page. Then, locate and click the name of the product on the Tivoli software information center page.

Information is organized by product and includes READMEs, installation guides, user's guides, administrator's guides, and developer's references.

Note: To ensure proper printing of PDF publications, select the **Fit to page** check box in the Adobe Acrobat Print window (which is available when you click **File → Print**).

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. After installation, you also can use the keyboard instead of the mouse to operate all features of the graphical user interface.

Contacting software support

Before contacting IBM Tivoli Software support with a problem, refer to the Tivoli Software support Web site at:

http://www.ibm.com/software/sysmgmt/products/support/

If you need additional help, contact software support by using the methods described in the *IBM Software Support Guide* at the following Web site:

http://techsupport.services.ibm.com/guides/handbook.html

The guide provides the following information:
- Registration and eligibility requirements for receiving support
- Telephone numbers and e-mail addresses, depending on the country in which you are located
- A list of information you should gather before contacting customer support

Conventions used in this book

This reference uses several conventions for special terms and actions and for operating system-dependent commands and paths.

Typeface conventions

The following typeface conventions are used in this reference:

Bold
: Lowercase commands or mixed case commands that are difficult to distinguish from surrounding text, keywords, parameters, options, names of Java classes, and objects are in **bold**.

Italic
: Titles of publications, and special words or phrases that are emphasized are in *italic*.

<*Italic*>
: Variables are set off with < > and are in <*italic*>.

`Monospace`
: Code examples, command lines, screen output, file and directory names that are difficult to distinguish from surrounding text, system messages, text that the user must type, and values for arguments or command options are in `monospace`.

Operating system differences

This book uses the UNIX® convention for specifying environment variables and for directory notation. When using the Windows® command line, replace *$variable* with *%variable%* for environment variables and replace each forward slash (/) with a backslash (\) in directory paths. If you are using the bash shell on a Windows system, you can use the UNIX conventions.

Chapter 1. IBM Tivoli Directory Server tuning general overview

This guide provides tuning information for IBM Tivoli Directory Server and the related IBM Database 2™ (DB2®) database. IBM Tivoli Directory Server is a Lightweight Directory Access Protocol (LDAP) directory that provides a layer on top of DB2, allowing users to efficiently organize, manipulate, and retrieve data stored in the DB2 database. Tuning for optimal performance is primarily a matter of adjusting the relationships between the LDAP server and DB2 according to the nature of your workload.

Because each workload is different, instead of providing exact values for tuning settings, guidelines are provided, where appropriate, for how to determine the best settings for your system. The measurements in this guide use the DirectoryMark (DirMark) 1.2 industry-standard benchmark provided by Mindcraft Inc., but the processes outlined can be applied to any workload. See Appendix A, "DirectoryMark 1.2", on page 47 and Appendix B, "Platform configurations", on page 49 for more information about DirMark 1.2 and the platform configurations used in the examples.

Attention: The measurements in this guide are from a previous release of IBM Directory Server. The processes shown are valid for IBM Tivoli Directory Server Version 5.2; however, your measurements might differ from those shown in the examples.

IBM Tivoli Directory Server 5.2 application components

The following figure illustrates how IBM Tivoli Directory Server components interact with each other. Tuning these components can result in improved performance.

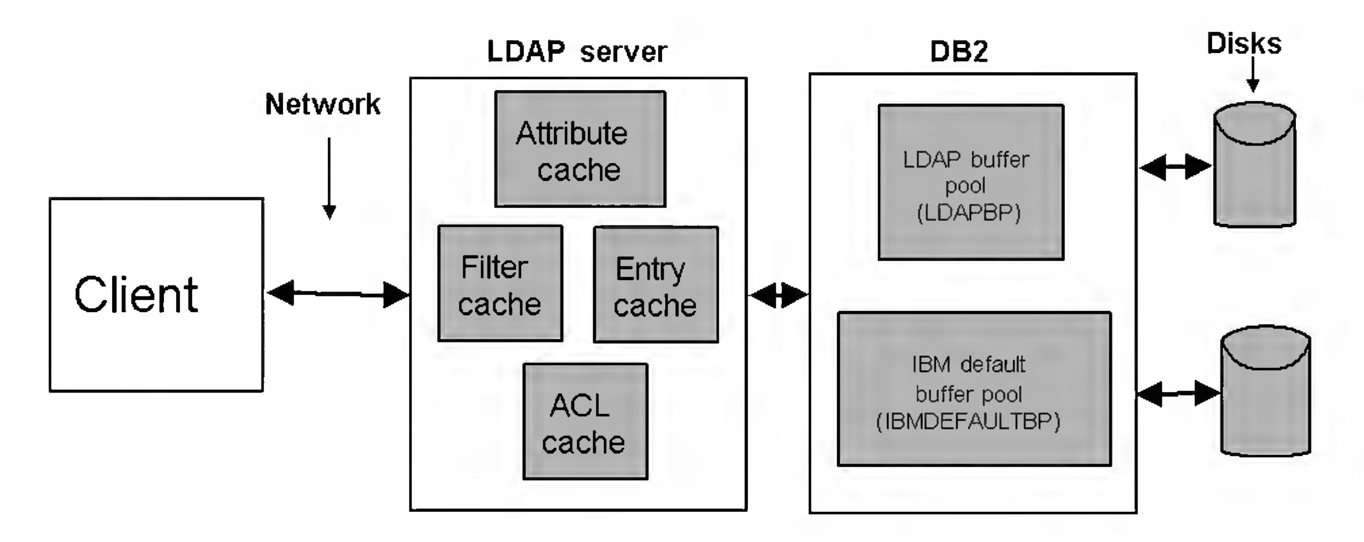


Figure 1. IBM Tivoli Directory Server 5.2

The arrows in Figure 1 on page 1 represent the path of a query issued from a client computer. The query follows a path from the IBM Tivoli Directory Server client to the LDAP server, to DB2, to the physical disks in search of entries that match the query's search filter settings. The shorter the path to matching entries, the better overall performance you can expect from your system.

For example, if a query locates all the matching entries in the LDAP server, access to DB2 and the disks is not necessary. If the matching entries are not found in the LDAP server, the query continues on to DB2 and, if necessary, to the physical disks as a last resort. Because of the time and resources it takes to retrieve data from disk, it is better from a performance standpoint to allocate a significant amount of memory to the LDAP server caches and DB2 buffer pools.

LDAP caches and DB2 buffer pools

Caches and buffer pools store previously retrieved data and can significantly improve performance by reducing disk access. When requested data is found within a cache or buffer pool, it is called a cache hit. A cache miss occurs when requested data is not located in a cache or buffer pool.

Because the type of information in each cache and buffer pool is different, it is useful to understand why and when each cache is accessed.

LDAP caches

There are four LDAP caches:

Attribute cache

The attribute cache stores configured attributes and their values in memory. When a search is performed using a filter that contains all cached attributes, and the filter is of a type supported by the attribute cache manager, the filter can be resolved in memory. Resolving filters in memory leads to improved search performance.

When the client issues a query for some data, the first place the query goes is the attribute cache. There are two things that can happen when a query arrives at the attribute cache:

- **All attributes used in the search filter are cached and the filter is of a type that can be resolved by the attribute cache manager.** If this is the case, the list of matching entry IDs is resolved in memory using the attribute cache manager.
- **Some or all of the attributes used in the search filter are not cached or the filter is of a type that cannot be resolved by the attribute cache manager.** If this is the case, the query is sent to the filter cache for further processing.

See "LDAP attribute cache" on page 5 for more information.

Filter cache

The filter cache contains cached entry IDs that match a search filter that was previously resolved in DB2. When the client issues a query for some data and that query cannot be resolved in memory by the attribute cache manager, the query goes to the filter cache. There are two things that can happen when a query arrives at the filter cache:

- **The IDs that match the filter settings used in the query are located in the filter cache.** If this is the case, the list of the matching entry IDs is sent to the entry cache.

- **The filter cache does not contain the matching entry IDs.** In this case, the query must access DB2 in search of the desired data.

See "LDAP filter cache" on page 6 for more information.

Entry cache

The entry cache contains cached entry data. Entry IDs are sent to the entry cache. If the entries that match the entry IDs are in the entry cache, then the results are returned to the client. If the entry cache does not contain the entries that correspond to the entry IDs, the query goes to DB2 in search of the matching entries.

See "Entry cache" on page 9 for more information.

ACL cache

The Access Control List (ACL) cache contains information about the access permissions of recently queried entries, such as the entry owner and whether the entry's permissions are explicit or inherited. Having this information cached in memory can speed up the process of determining whether the user who submitted the query is authorized to see all, some, or none of its results.

DB2 buffer pools

There are two DB2 buffer pools:

LDAPBP

LDAPBP contains cached entry data (ldap_entry) and all of the associated indexes. LDAPBP is similar to the entry cache, except that LDAPBP uses different algorithms in determining which entries are cached. It is possible that an entry that is not cached in the entry cache is located in LDAPBP. If the requested data is not found in the entry cache or LDAPBP, the query must access the physical disks.

See "LDAPBP buffer pool size" on page 18 for more information.

IBMDEFAULTBP

DB2 system information, including system tables and other LDAP information, is cached in the IBMDEFAULTBP. You might need to adjust the IBMDEFAULTBP cache settings for better performance. See "IBMDEFAULTBP buffer pool size" on page 19 for more information.

Memory allocation between LDAP caches and buffer pools

The LDAP caches are generally more efficient as a means of caching LDAP searches; however, parts of the LDAP cache get invalidated on updates and must be reloaded before performance benefits return. Some experimentation between the two caching schemes is required to find the best memory allocation for your workload.

IBM Tivoli Directory Server tuning overview

Tuning the LDAP server can significantly improve performance by storing useful data in the caches. It is important to remember, however, that tuning the LDAP server alone is insufficient. Some tuning of DB2 is also required for optimal performance.

The most significant performance tuning related to the IBM Tivoli Directory Server involves the LDAP caches. LDAP caches are fast storage buffers in memory used to store LDAP information such as queries, answers, and user authentication for future use. While LDAP caches are useful mostly for applications that frequently

retrieve repeated cached information, they can greatly improve performance by avoiding calls to the database. See "LDAP caches" on page 5 for information about how to tune the LDAP caches.

DB2 tuning overview

DB2 serves as the data storage component of the IBM Tivoli Directory Server. Tuning DB2 results in overall improved performance.

This guide contains several recommendations for tuning DB2, but the most commonly tuned items are:

- **DB2 buffer pools** – Buffer pools are DB2 data caches. Each buffer pool is a data cache between the applications and the physical database files. Adjusting the size of the DB2 buffer pools can result in improved performance. See "DB2 buffer pool tuning" on page 18 for information about buffer pool tuning.
- **Optimization and organization** – After initially loading a directory, or after a number of updates have been performed, it is very important to update database statistics and organization for DB2 to perform optimally. See "Optimization and organization (reorgchk and reorg)" on page 20 for more information.
- **Indexes** –Indexes can make locating data on disk very fast, providing a significant boost to performance. For information about how to create indexes, see "Indexes" on page 25.

Attention: You must place the DB2 log on a physical disk drive separate from the data. While there might be some performance benefit to having the DB2 log and data on different drives, data-integrity concerns require the separation. Use the following command to set the path to the DB2 log file directory:

```
UPDATE DATABASE CONFIGURATION FOR database_alias USING NEWLOGPATH path
```

Be sure the database instance owner has write access to the specified path or the command fails.

Generic LDAP application tips

The following are some tips that can help improve performance:

- Perform searches on indexed attributes only. See "Indexes" on page 25 for instructions for defining and verifying indexes for IBM Tivoli Directory Server.
- Open a connection only once and reuse it for many operations if possible.
- Minimize the number of searches by retrieving multiple attribute values at one time.
- Put frequently searched attributes in the attribute cache.
- Retrieve only the attributes you need. Do not use ALL by default. For example, when you search for the groups a user belongs to, ask for only the Distinguished Names (DNs), and not the entire group.
- Minimize and batch updates (**add**, **modify**, **modrdn**, **delete**) when possible.

Chapter 2. IBM Tivoli Directory Server tuning

This chapter discusses the following performance tuning tasks for the IBM Tivoli Directory Server:
- Tuning LDAP caches
- Determining how directory size affects performance

LDAP caches

LDAP caches are fast storage buffers in memory used to store LDAP information such as queries, answers, and user authentication for future use. Tuning the LDAP caches is crucial to improving performance.

An LDAP search that accesses the LDAP cache can be faster than one that requires a connection to DB2, even if the information is cached in DB2. For this reason, tuning LDAP caches can improve performance by avoiding calls to the database. The LDAP caches are especially useful for applications that frequently retrieve repeated cached information. See Figure 1 on page 1 for an illustration of the LDAP caches.

The following sections discuss each of the LDAP caches and demonstrate how to determine and set the best cache settings for your system. The examples used in the tables and measurements are based on DirMark 1.2 data. Keep in mind that every workload is different, and some experimentation will likely be required in order to find the best settings for your workload.

Note: Cache sizes for the filter cache, ACL cache, and entry cache are measured in numbers of entries.

LDAP attribute cache

The attribute cache stores configured attributes and their values in memory. When a search is performed using a filter that contains all cached attributes, and the filter is of a type supported by the attribute cache manager, the filter can be resolved in memory. Resolving filters in memory leads to improved search performance.

When the client issues a query for some data, the first place the query goes is the attribute cache. There are two things that can happen when a query arrives at the attribute cache:

- **All attributes used in the search filter are cached and the filter is of a type that can be resolved by the attribute cache manager.** If this is the case, the list of matching entry IDs is resolved in memory using the attribute cache manager.

  The attribute cache manager can resolve simple filters of the following types:
  - exact match filters
  - presence filters

  The attribute cache manager can resolve complex filters only if they are conjunctive. In addition, the subfilters within the complex filters must be of the following types:
  - exact match filters
  - presence filters

– conjunctive filters

Filters containing attributes with language tags are not resolved by the attribute cache manager.

For example, if the attributes objectclass, uid, and cn are all cached, the following filters can be resolved in memory within the attribute cache manager:
- `(cn=Karla)`
- `(cn=*)`
- `(&(objectclass=eperson)(cn=Karla))`
- `(&(objectclass=eperson)(cn=*)(uid=1234567))`
- `(&(&(objectclass=eperson)(cn=*))(uid=1234567))`
- `(&(uid=1234567)(&(objectclass=eperson)(cn=*)))`

- **Either some or all of the attributes used in the search filter are not cached or the filter is of a type that cannot be resolved by the attribute cache manager.** If this is the case, the query is sent to the filter cache for further processing.

  Note: If there are no attributes in the attribute cache, the attribute cache manager determines this quickly, and the query is sent to the filter cache.

  For example, if the attributes objectclass, uid, and cn are the only cached attributes, the following filters will not be able to be resolved in memory by the attribute cache manager:
  - `(sn=Smith)`
  - `(cn=K*)`
  - `(|(objectclass=eperson)(cn~=Karla))`
  - `(&(objectclass=eperson)(cn=K*)(uid=1234567))`
  - `(&(&(objectclass=eperson)(cn<=Karla))(uid=1234567))`
  - `(&(uid=1234567)(&(objectclass=eperson)(sn=*)))`

Determining which attributes to cache

To determine which attributes to cache, experiment with adding some or all of the attributes listed in the **cached_attribute_candidate_hit** attribute to the attribute cache. Then run your workload and measure the differences in operations per second. For information about the **cached_attribute_candidate_hit** attribute, see "ldapsearch with "cn=monitor"" on page 35.

LDAP filter cache

When the client issues a query for data and the query cannot be resolved in memory by the attribute cache manager, the query goes to the filter cache. This cache contains cached entry IDs. There are two things that can happen when a query arrives at the filter cache:

- **The IDs that match the filter settings used in the query are located in the filter cache.** If this is the case, the list of the matching entry IDs is sent to the entry cache.
- **The matching entry IDs are not cached in the filter cache.** In this case, the query must access DB2 in search of the desired data.

Filter cache size

To determine how big your filter cache should be, run your workload with the filter cache set to different values and measure the differences in operations per

second. For example, Figure 2 shows varying operations per second based on different filter cache sizes for one installation:

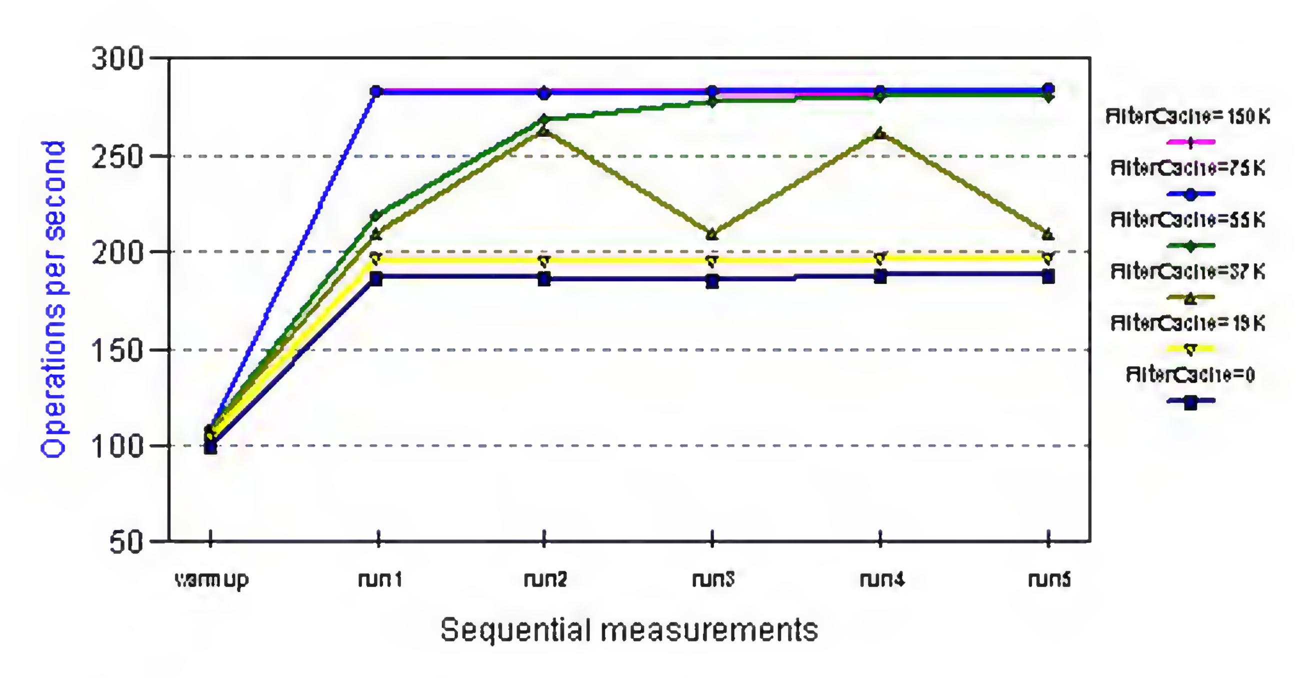


Figure 2. Varying the size of the filter cache

From the results in Figure 2, it can be concluded that for this workload, there is little benefit to a filter cache size of 20,000 over no filter cache at all. Sizes in the 20,000–50,000 range result in unpredictable varying behaviors as some queries hit in the cache while others miss.

For this workload it appears that a filter cache large enough to hold 75,000 entries results in the best performance. There is no benefit in making the filter cache any larger than this. The following table shows that setting the filter cache at 75,000 results in 50% more address lookup operations per second than if the filter cache was set to zero:

Table 1. Effects of varying the size of the filter cache

Filter cache size	75,000	0
Filter cache bypass limit	100	100
Entry cache size	460,000	460,000
Address Lookup (operations per second)	283	188
Address Lookup warmup (operations per second)	108	101

See "LDAP cache configuration variables" on page 11 to set the filter cache size.

Filter cache size with updates

Figure 3 shows that, for the test installation, there is no performance benefit in allocating any memory to the filter cache if even a small fraction of the operations in the workload are updates.

If this proves to be the case for your workload, the only way to retain the performance advantage of a filter cache when updates are involved is to batch your updates. This allows long intervals during which there are only searches. If you cannot batch updates, specify a filter cache size of zero and allot more memory to other caches and buffer pools. See "LDAP cache configuration variables" on page 11 for instructions on how to set configuration variables such as filter cache size.

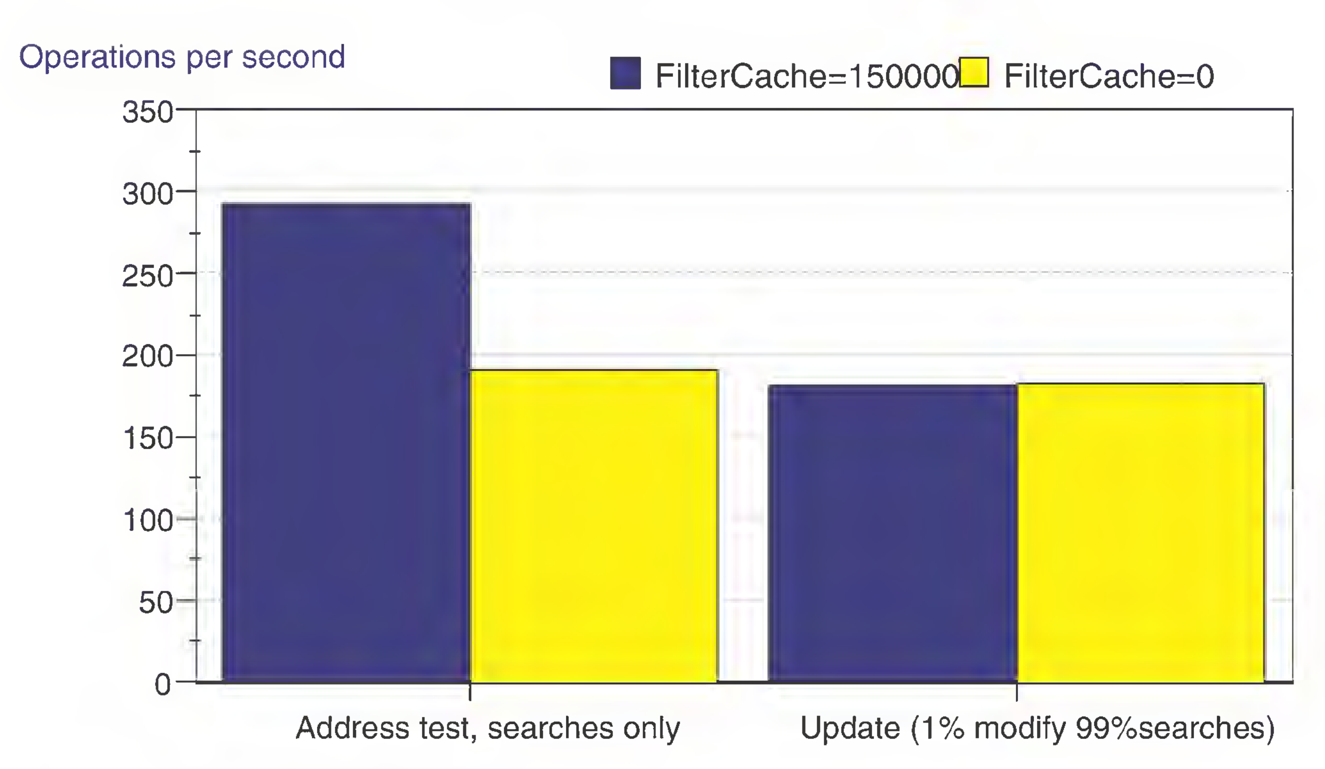


Figure 3. Effect of updates on the performance of the filter cache

Filter cache bypass limits

The filter cache bypass limit configuration variable limits the number of entries that can be added to the filter cache. For example, if the bypass limit variable is set to 1,000, search filters that match more than 1,000 entries are not added to the filter cache. This prevents large, uncommon searches from overwriting useful cache entries. To determine the best filter cache bypass limit for your workload, run your workload repeatedly and measure the throughput.

For example, Figure 4 on page 9 shows operations per second based on varying cache bypass limit sizes.

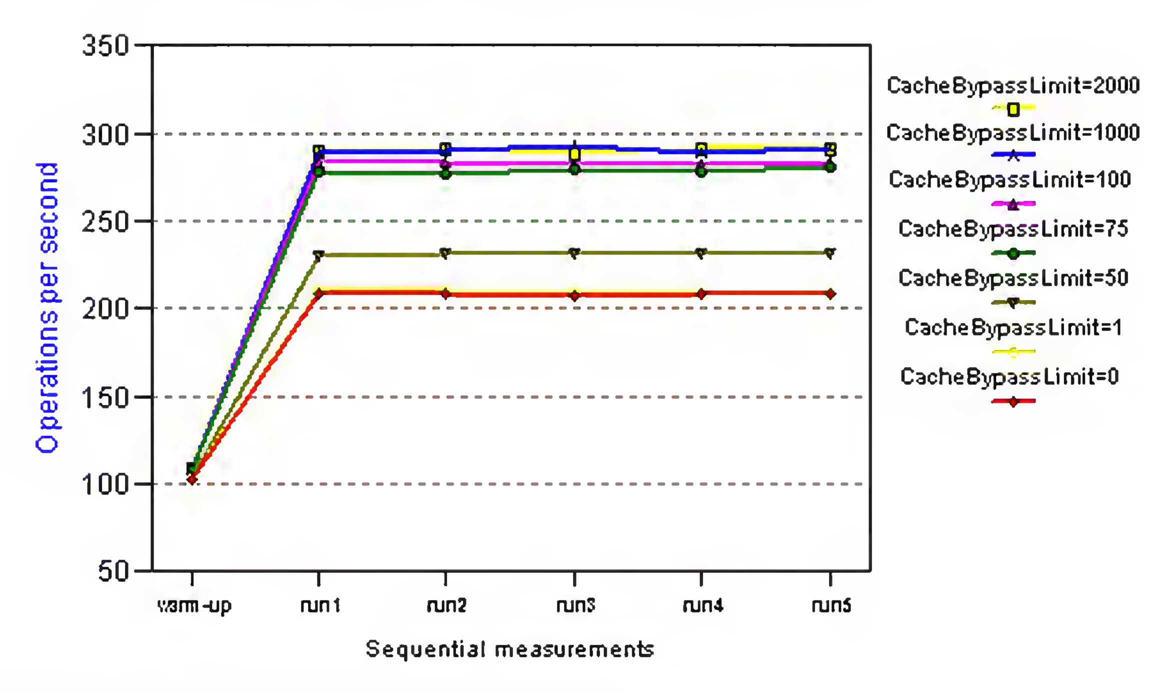


Figure 4. Varying the filter cache bypass limit

For the workload in Figure 4, setting the limit too low downgrades performance by preventing valuable filters from being cached. Setting the filter bypass limit to approximately 100 appears to be the best size for this workload. Setting it any larger benefits performance only slightly.

See "LDAP cache configuration variables" on page 11 to set the filter cache bypass limit.

Entry cache

The entry cache contains cached entry data. Entry IDs are sent to the entry cache. If the entries that match the entry IDs are in the entry cache, then the results are returned to the client. If the entry cache does not contain the entries that correspond to the entry IDs, the query goes to DB2 in search of the matching entries.

Entry cache size

To determine how big your entry cache should be, run your workload with the entry cache set to different sizes and measure the differences in operations per second. For example, Figure 5 on page 10 shows varying operations per second based on different entry cache sizes:

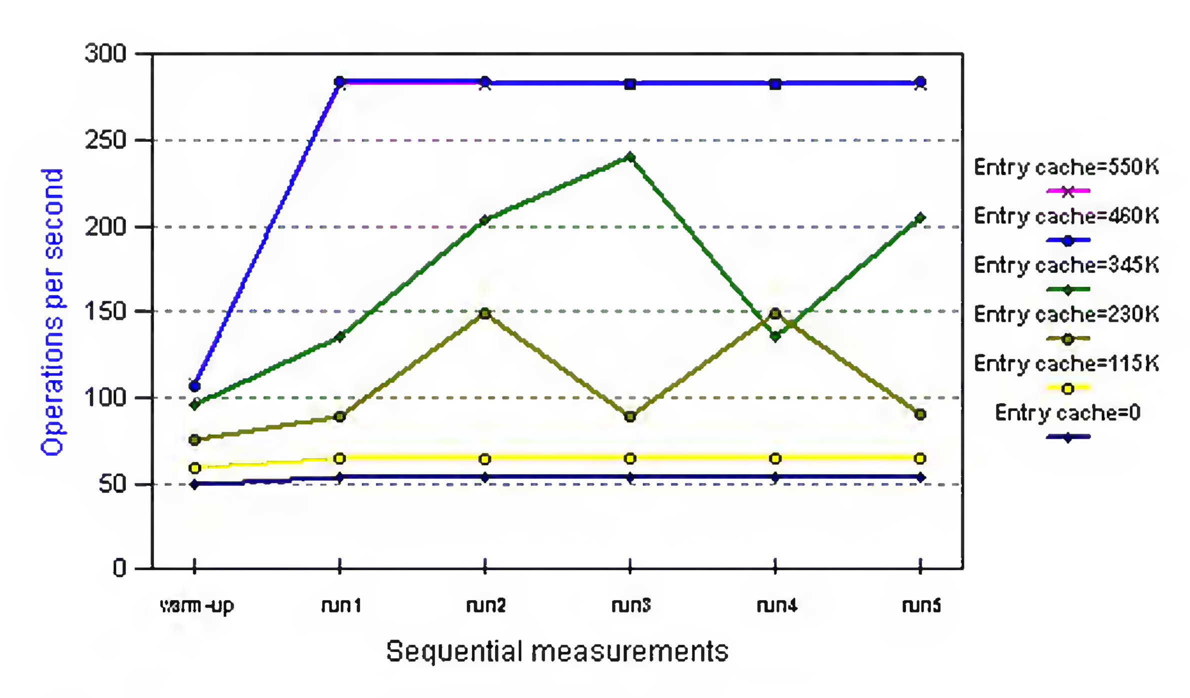


Figure 5. Varying the size of the entry cache

From the results in Figure 5, it can be concluded that for this workload, there is little benefit to an entry cache size of 100,000 over no entry cache at all. Sizes in the 200,000–400,000 range result in unpredictable varying behaviors as some queries hit in the cache while others miss.

For this workload it appears that an entry cache large enough to hold 460,000 entries results in the best performance. There is no benefit to making the entry cache any larger than this. Setting the entry cache at 460,000 results in 5 times as many operations per second than if entry cache was set to zero:

Table 2. Effects of varying the size of the entry cache

Entry cache size	**460,000**	**0**
Filter cache size	100	100
Filter cache bypass limit	150,000	150,000
Address Lookup (operations per second)	284	54
Address Lookup warmup (operations per second)	107	50

To find the best cache size for your workload, you must run your workload with different cache sizes. See "LDAP cache configuration variables" on page 11 to set the filter cache size.

Measuring cache entry sizes

Filter cache and entry cache sizes are measured in numbers of entries. When determining how much memory to allocate to your LDAP caches, it can be useful to know how big the entries in your cache are.

The following example shows how to measure the size of cached entries:

Note: This example calculates the average size of an entry in a sample entry cache, but the average filter cache entry size can be calculated similarly.

1. From the LDAP server:
   a. Set the filter cache size to zero.
   b. Set the entry cache size to a small value; for example, 200.
   c. Start **ibmslapd**.
2. From the client:
   a. Run your application.
   b. Find the entry cache population (call this *population1*) using the following command:
      ```
      ldapsearch -h servername -s base -b cn=monitor objectclass=* | grep
       entry_cache_current
      ```
3. From the LDAP Server:
   a. Find the memory used by **ibmslapd** (call this *ibmslapd1*):
      - On AIX® operating systems, use `ps v`
      - On Windows operating systems, use the **VM size** column in the **Task Manager**.
   b. Stop **ibmslapd**.
   c. Increase the size of the entry cache but keep it smaller than your working set.
   d. Start **ibmslapd**.
4. Run your application again and find the entry cache population (call this *population2*). See step 2b for the command syntax.
5. Find the memory used by **ibmslapd** (call this *ibmslapd2*). See step 3a for the command syntax.
6. Calculate the size of an entry cache entry using the following formula:
   ```
   (ibmslapd size2 - ibmslapd size1)  /
   (entry cache population2 - entry cache population1)
   ```

For example, using this formula with the 500,000-entry database used by DirMark 1.2 results in the following measurement:

```
(192084 KB – 51736 KB) / (48485 – 10003) = 3.65 KB per entry
```

LDAP cache configuration variables

LDAP cache configuration variables allow you to set the LDAP cache sizes, bypass limits, and other variables that affect performance.

Configuring attribute caching

You can configure attribute caching for the directory database, the changelog database, or both. Typically, there is no benefit from configuring attribute caching for the changelog database unless you perform very frequent searches of the changelog.

Using the Web Administration Tool

To configure the attribute cache using the Web Administration Tool:

1. Expand the **Manage server properties** category in the navigation area of the Web Administration Tool and select the **Attribute cache** tab.
2. You can change the amount of memory in bytes available to the directory attribute cache by changing the **Directory cached attribute size (in kilobytes)** field. The default is 16,384 KB (16 MB).
3. You can change the amount of memory in bytes available to the changelog attribute cache by changing the **Changelog cached attribute size (in kilobytes)** field. The default is 16,384 KB (16 MB).

   Note: This selection is disabled if a changelog has not been configured.

To add attributes to the attribute cache:

4. Select the attribute that you want to add as a cached attribute from the **Available attributes** menu. Only those attributes that can be cached are displayed in this menu; for example, sn.

   Note: An attribute remains in the list of available attributes until it has been placed in both the cn=directory and the cn=changelog containers.
5. Click either **Add to cn=directory** or **Add to cn=changelog**. The attribute is displayed in the appropriate list box. You can list the same attribute in both containers.

   Note: **Add to cn=changelog** is disabled if a changelog has not been configured.
6. Repeat this process for each attribute you want to add to the attribute cache.
7. When you are finished, click **Apply** to save your changes without exiting, or click **OK** to apply your changes and exit, or click **Cancel** to exit this panel without making any changes.

Using the command line

To configure the attribute cache through the command line, issue the following command:

```
ldapmodify -D <adminDN> -w<adminPW> -i<filename>
```

where *<filename>* contains the following, for example.

- For the directory database:

  ```
  dn: cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, cn=Configuration
  changetype: modify
  add: ibm-slapdCachedAttribute
  ibm-slapdCachedAttribute:  sn
  -
  add: ibm-slapdCachedAttribute
  ibm-slapdCachedAttribute:  cn
  -
  add: ibm-SlapdCachedAttributeSize
  ibm-SlapdCachedAttributeSize: 16384
  ```
- For the changelog database:

  ```
  dn: cn=change log, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, cn=Configuration
  changetype: modify
  add: ibm-slapdCachedAttribute
  ibm-slapdCachedAttribute:  changetype
  -
  add: ibm-SlapdCachedAttributeSize
  ibm-SlapdCachedAttributeSize: 16384
  ```

See the *IBM Tivoli Directory Server Version 5.2 Administration Guide* for more information.

Setting other LDAP cache configuration variables

You can set LDAP configuration variables using the Web Administration Tool or the command line.

Using the Web Administration Tool

To set LDAP configuration variables using the Web Administration Tool:

1. Expand the **Manage server properties** category in the navigation area of the Web Administration tool.
2. Click **Performance**.
3. You can modify any of the following configuration variables:
   - **Cache ACL information** — This option must be selected for the **Maximum number of elements in ACL cache** settings to take effect.
   - **Maximum number of elements in ACL cache** (ACL cache size) — The default is 25,000.
   - **Maximum number of elements in entry cache** (entry cache size) — Specify the maximum number of elements in the entry cache. The default is 25,000. See "Entry cache" on page 9 for more information about the entry cache.
   - **Maximum number of elements in search filter cache** (filter cache size)— The search filter cache consists of the requested search filters and resulting entry identifiers that matched. On an update operation, all filter cache entries are invalidated. The default is 25,000. See "LDAP filter cache" on page 6 for more information about the filter cache.
   - **Maximum number of elements from a single search added to search filter cache** (filter cache bypass limit) — If you select **Elements**, you must enter a number. The default is 100. Otherwise select **Unlimited**. Search filters that match more entries than the number specified here are not added to the search filter cache. See "Filter cache bypass limits" on page 8 for more information about bypass limits.
4. When you are finished, click **OK** to apply your changes, or click **Cancel** to exit the panel without making any changes.

Using the command line

To set LDAP configuration variables using the command line, issue the following command:

```
ldapmodify -DAdminDN -wAdminpassword  -ifilename
```

where the file *filename* contains:

```
dn: cn=Directory,cn=RDBM Backends,cn=IBM Directory,cn=Schemas,cn=Configuration
changetype: modify
replace: ibm-slapdDbConnections
ibm-slapdDbConnections:  15

dn: cn=Front End, cn=Configuration
changetype: modify
replace: ibm-slapdACLCache
ibm-slapdACLCache: TRUE
-
replace: ibm-slapdACLCacheSize
ibm-slapdACLCacheSize: 25000
-
```

```
replace: ibm-slapdEntryCacheSize
ibm-slapdEntryCacheSize: 25000
-
replace: ibm-slapdFilterCacheSize
ibm-slapdFilterCacheSize: 25000
-
replace: ibm-slapdFilterCacheBypassLimit
ibm-slapdFilterCacheBypassLimit: 100
```

Additional settings

There are several additional settings that affect performance by putting limits on client activity, minimizing the impact to server throughput and resource usage, such as:

- ibm-slapdSizeLimit: 500
- ibm-slapdTimeLimit: 900
- ibm-slapdIdleTimeOut: 300
- ibm-slapdMaxEventsPerConnection: 100
- ibm-slapdMaxEventsTotal: 0
- ibm-slapdMaxNumOfTransactions: 20
- ibm-slapdMaxOpPerTransaction: 5
- ibm-slapdMaxTimeLimitOfTransactions: 300
- ibm-slapdPagedResAllowNonAdmin: TRUE
- ibm-slapdPagedResLmt: 3
- ibm-slapdSortKeyLimit: 3
- ibm-slapdSortSrchAllowNonAdmin: TRUE

Note: Default values are shown.

Directory size

It is possible to measure how the size of your directory impacts performance by running your workload with different directory sizes. For example, if your current directory contains 500,000 entries, try tuning your workload with a 100,000 entry directory and a 1,000,000 entry directory.

It is important when you run your workload that you consider several measurements. For example, measuring the number of operations per second as shown in Figure 6 on page 15, it appears that performance degrades significantly as the database size grows.

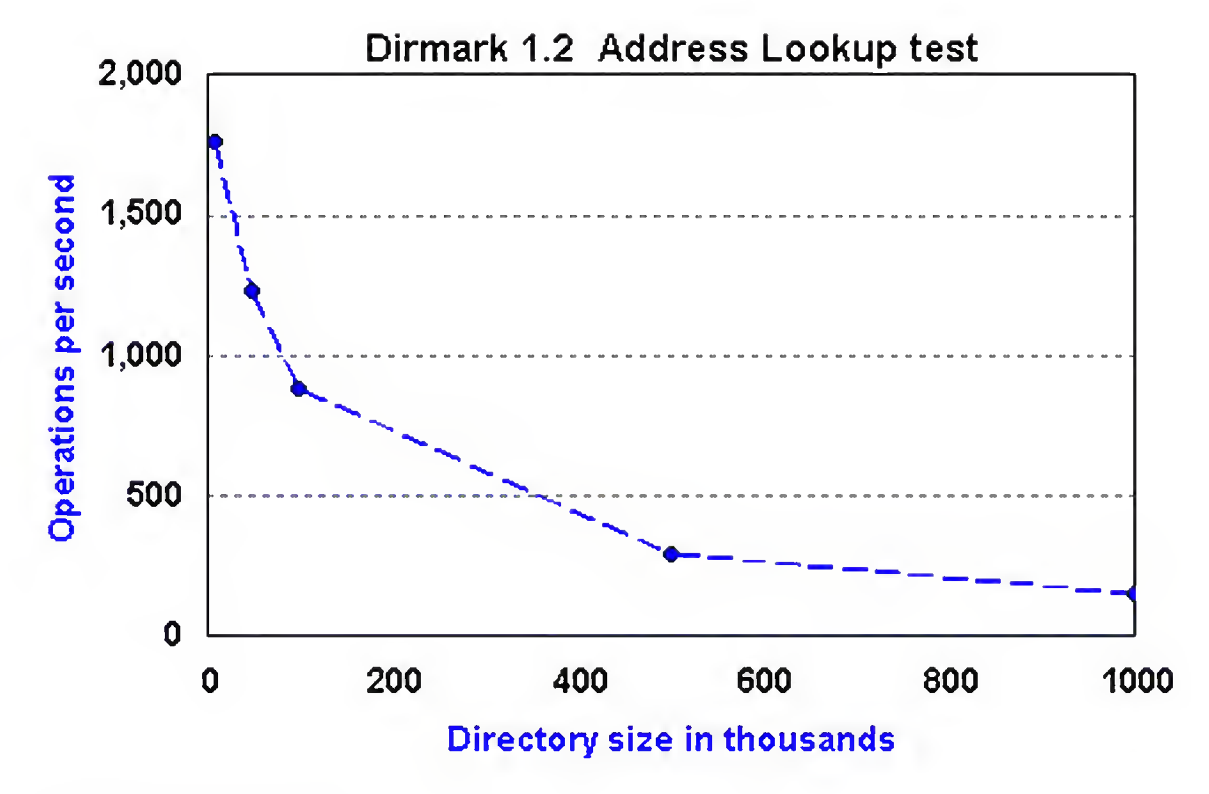


Figure 6. Operations per second

However, the DirMark Address Lookup test includes a large fraction of wildcard searches and exact-match searches, such as "(sn=Smith)" that return all entries where the sn value is "Smith". Both of these types of searches typically return multiple entries in response to a single search request. As Figure 7 on page 16 shows, as the size of the directory grows, so does the number of entries returned in response to wildcard and exact-match search requests.

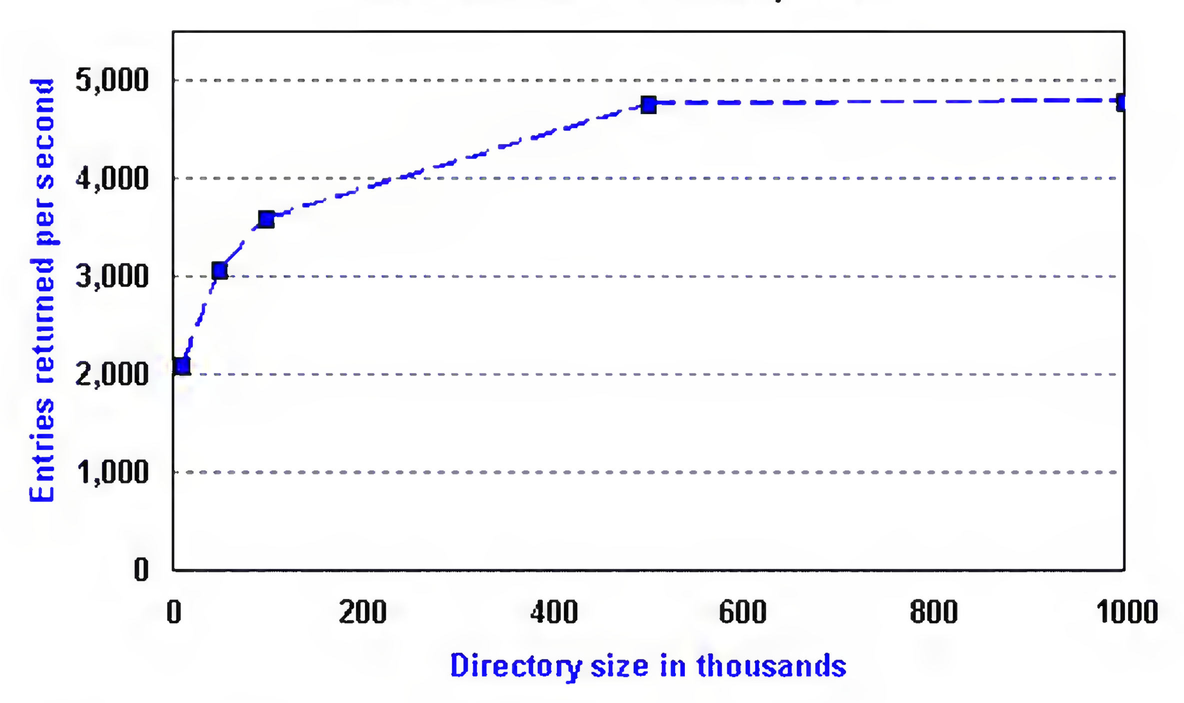


Figure 7. Entries returned per second

In this situation, the number of entries returned per second is a truer measure of throughput than operations per second, because each operation requires more work to be performed as the size of the database grows.

Note: As your directory grows, it might become necessary to readjust the sizes of the LDAP caches and DB2 buffer pools. You can determine the optimal sizes for your caches and buffer pools using the guidelines in "LDAP caches" on page 5 and "DB2 buffer pool tuning" on page 18.

Chapter 3. DB2 tuning

IBM Tivoli Directory Server uses DB2 as the data store and Structured Query Language (SQL) as the query retrieval mechanism. While the LDAP server caches LDAP queries, answers, and authentication information, DB2 caches tables, indexes, and statements.

Many DB2 configuration parameters affect either the memory (buffer pools) or disk resources. Since disk access is usually much slower than memory access, the key database performance tuning objective is to decrease the amount of disk activity.

This chapter covers the following types of DB2 tuning:
- DB2 buffer pool tuning
- Optimization and organization (**reorgchk** and **reorg**)
- Other DB2 configuration parameters
- Backing up and restoring the database (**backup** and **restore**)

For detailed information about DB2 commands, see the DB2 documentation at the following Web site: http://www.ibm.com/software/data/db2/library/

Attention: Only users listed as database administrators can run the DB2 commands. Be sure the user ID running the DB2 commands is a user in the dbsysadm group (UNIX operating systems) or a member of the Administrator group (Windows operating systems.) This includes the DB2 instance owner and root.

If you have any trouble running the DB2 commands, check to ensure that the DB2 environment variables have been established by running **db2profile** (if not, the **db2 get** and **db2 update** commands will not work). The script file **db2profile** is located in the sqllib subdirectory under the instance owner's home directory. If you need to tailor this file, follow the comments inside the file to set your instance name, user paths, and default database name (the default path is /home/ldapdb2/sqllib/db2profile.) It is assumed that the user is logged in as **ibm-slapdDbUserId**. If logged in as the root user on a UNIX operating system, it is possible to switch to the instance owner as follows:

```
su - instance_owner
```

where *instance_owner* is the defined owner of the LDAP database.

To log on as the database administrator on a Windows 2000 operating system, run the following command:

```
runas /user:instance_owner db2cmd
```

where *instance_owner* is the defined owner of the LDAP database.

Note: If you have problems connecting to the database on Windows systems, check the DB2INSTANCE environment variable. By default this variable is set to `DB2`. However, to connect to the database, the environment variable must be set to the database instance name.

For additional stability and performance enhancements, upgrade to the latest version of DB2.

DB2 buffer pool tuning

DB2 buffer pool tuning is one of the most significant types of DB2 performance tuning. A buffer pool is a data cache between LDAP and the physical DB2 database files for both tables and indexes. DB2 buffer pools are searched when entries and their attributes are not found in the entry cache. Buffer pool tuning typically needs to be done when the database is initially loaded and when the database size changes significantly.

There are several considerations to keep in mind when tuning the DB2 buffer pools; for example:

- If there are no buffer pools, all database activity results in disk access.
- If the size of each buffer pool is too small, LDAP must wait for DB2 disk activity to satisfy DB2 SQL requests.
- If one or more buffer pools is too large, memory on the LDAP server might be wasted.
- If the total amount of space used by the LDAP caches and both buffer pools is larger than physical memory available on the server, operating system paging (disk activity) will occur.

To get the current DB2 buffer pool sizes, run the following commands:

```
db2 connect to database_name
db2 "select bpname,npages,pagesize from sysibm.sysbufferpools"
```

where *database_name* is the name of the database.

The following example output shows the default settings for the example above:

```
BPNAME                NPAGES       PAGESIZE
------------------  -----------  -----------
IBMDEFAULTBP              29500         4096
LDAPBP                    1230         32768

  2 record(s) selected.
```

Buffer pool sizes

In IBM Tivoli Directory Server 5.2, the LDAP directory database (DB2) has two buffer pools: LDAPBP and IBMDEFAULTBP. The size of each buffer pool needs to be set separately, but the method for determining how big each should be is the same: Run your workload with the buffer pool sizes set to different values and measure the differences in operations per second.

Note: DB2 does not allow buffer pools to be set to zero.

LDAPBP buffer pool size

This buffer pool contains cached entry data (ldap_entry) and all of the associated indexes. LDAPBP is similar to the entry cache, except that LDAPBP uses different algorithms in determining which entries to cache. It is possible that an entry that is not cached in the entry cache is located in LDAPBP.

To determine the best size for your LDAPBP buffer pool, run your workload with the LDAPBP buffer pool size set to different values and measure the differences in

operations per second. For example, Figure 8 shows varying operations per second based on different LDAPBP buffer pool sizes:

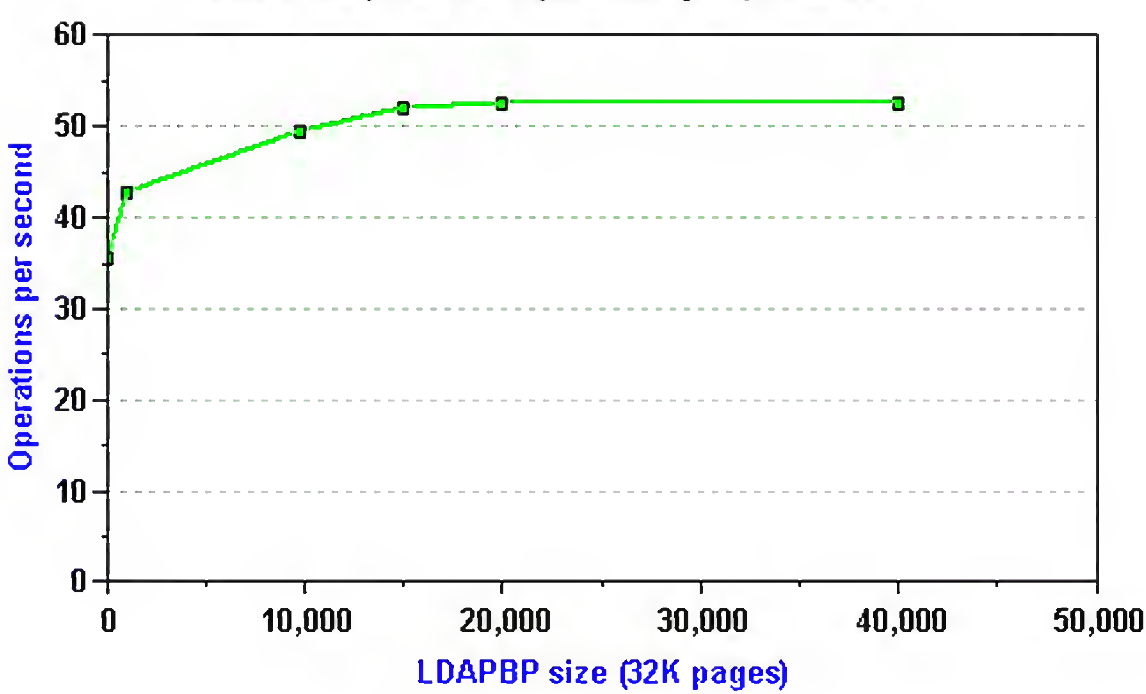


Figure 8. Varying the size of LDAPBP

For the workload in the above example, the best performance results from a LDAPBP size of approximately 15,000 32K pages. However, the performance gain of 15,000 over a size of 9,800 is slight. In a memory-constrained environment, setting the LDAPBP size to 9,800 saves approximately 166 MB of memory.

IBMDEFAULTBP buffer pool size

DB2 system information, including system tables and other information that is useful in resolving filters, is cached in the IBMDEFAULTBP buffer pool. You might need to adjust the IBMDEFAULTBP cache settings for better performance in the LDAPBP.

To determine the best size for your IBMDEFAULTBP buffer pool, run your workload with the buffer pool sizes set to different values and measure the differences in operations per second. For example, Figure 9 on page 20 shows varying operations per second based on different IBMDEFAULTBP buffer pool sizes:

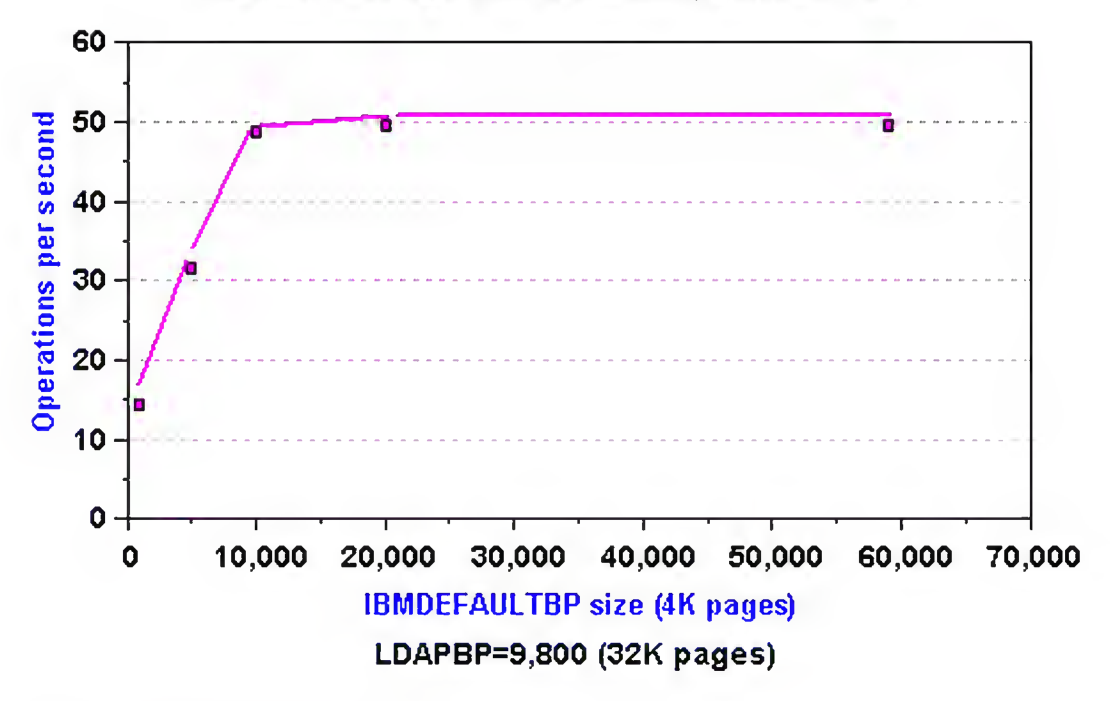


Figure 9. Varying the size of IBMDEFAULTBP

For the workload in the above example, setting the IBMDEFAULTBP large enough to hold the working set improves throughput more than three times over a small buffer pool size. There is little additional benefit to setting IBMDEFAULTBP larger than 20,000 4K pages.

Setting buffer pool sizes

Use the `alter bufferpool` command to set the IBMDEFAULTBP and LDAPBP buffer pool sizes. The following example shows the IBMDEFAULTBP and LDAPBP buffer pools being set:

```
db2 alter bufferpool ibmdefaultbp size 20000
db2 alter bufferpool ldapbp size 9800
db2 force applications all
db2stop
db2start
```

Optimization and organization (reorgchk and reorg)

DB2 uses a sophisticated set of algorithms to optimize the access to data stored in a database. These algorithms depend upon many factors, including the organization of the data in the database, and the distribution of that data in each table. Distribution of data is represented by a set of statistics maintained by the database manager.

In addition, IBM Tivoli Directory Server creates a number of indexes for tables in the database. These indexes are used to minimize the data accessed in order to locate a particular row in a table.

In a read-only environment, the distribution of the data changes very little. However, with updates and additions to the database, it is not uncommon for the

distribution of the data to change significantly. Similarly, it is quite possible for data in tables to become ordered in an inefficient manner.

To remedy these situations, DB2 provides tools to help optimize the access to data by updating the statistics and to reorganize the data within the tables of the database.

Optimization

Optimizing the database updates statistics related to the data tables, which improves performance and query speed. Optimize the database periodically or after heavy database updates (for example, after importing database entries). The **Optimize database** task in the IBM Tivoli Directory Server Configuration Tool uses the DB2 **runstats** command to update statistical information used by the query optimizer for all the LDAP tables.

Note: The **reorgchk** command also updates statistics. If you are planning to do a **reorgchk**, optimizing the database is unnecessary. See "Database organization (reorgchk and reorg)" for more information about the **reorgchk** command.

To optimize the database using the Configuration Tool:

1. Start the Configuration Tool by typing **ldapxcfg** on the command line.
2. Click **Optimize database** on the left side of the window.
3. On the Optimize database window, click **Optimize**.

After a message displays indicating the database was successfully optimized, you must restart the server for the changes to take effect.

To optimize the database using the command line, run the following command:

```
DB2 RUNSTATS ON TABLE table_name AND  DETAILED INDEXES ALL SHRLEVEL REFERENCE
```

Run the following commands for more detailed lists of runstats that might improve performance:

```
DB2 RUNSTATS ON TABLE table_name WITH DISTRIBUTION AND DETAILED INDEXES ALL SHRLEVEL
 REFERENCE
DB2 RUNSTATS ON TABLE ldapdb2.objectclass WITH DISTRIBUTION AND DETAILED INDEXES
 ALL SHRLEVEL REFERENCE
```

where *table_name* is the name of the table. You can use ALL for all tables.

Database organization (reorgchk and reorg)

Tuning the organization of the data in DB2 using the **reorgchk** and **reorg** commands is important for optimal performance.

The **reorgchk** command updates statistical information to the DB2 optimizer to improve performance, and reports statistics on the organization of the database tables.

The **reorg** command, using the data generated by **reorgchk**, reorganizes table spaces to improve access performance and reorganizes indexes so that they are more efficiently clustered. The **reorgchk** and **reorg** commands can improve both search and update operation performance.

Note: Tuning organizes the data on disk in a sorted order. Sorting the data on disk is beneficial only when accesses occur in a sorted order, which is not typically the case. For this reason, organizing the table data on disk typically yields little change in performance.

Performing a reorgchk

After a number of updates have been performed against DB2, table indexes can become sub-optimal and performance can degrade. Correct this situation by performing a DB2 **reorgchk** as follows:

```
db2 connect to ldapdb2
db2 reorgchk update statistics on table all
```

Where *ldapdb2* is the name of your database.

To generate a **reorgchk** output file (recommended if you plan to run the **reorg** command) add the name of the file to the end of the command, for example:

```
db2 reorgchk update statistics on table all > reorgchk.out
```

The following is a sample **reorgchk** report:

```
db2 => reorgchk current statistics on table all

Table statistics:

F1: 100 * OVERFLOW / CARD < 5
F2: 100 * TSIZE / ((FPAGES-1) * (TABLEPAGESIZE-76)) > 70
F3: 100 * NPAGES / FPAGES > 80

CREATOR   NAME                  CARD    OV    NP    FP    TSIZE  F1   F2 F3 REORG

--------------------------------------------------------------------------------

LDAPDB2   ACLPERM                  2     0     1     1      138   0    - 100 ---

LDAPDB2   ACLPROP                  2     0     1     1       40   0    - 100 ---

LDAPDB2   ALIASEDOBJECT            -     -     -     -        -   -    -   - ---

LDAPDB2   AUDIT                    1     0     1     1       18   0    - 100 ---

LDAPDB2   AUDITADD                 1     0     1     1       18   0    - 100 ---

LDAPDB2   AUDITBIND                1     0     1     1       18   0    - 100 ---

LDAPDB2   AUDITDELETE              1     0     1     1       18   0    - 100 ---

LDAPDB2   AUDITEXTOPEVENT          1     0     1     1       18   0    - 100 ---

LDAPDB2   AUDITFAILEDOPONLY        1     0     1     1       18   0    - 100 ---

LDAPDB2   AUDITLOG                 1     0     1     1       77   0    - 100 ---

...

SYSIBM    SYSINDEXCOLUSE         480     0     6     6    22560   0 100 100 ---

SYSIBM    SYSINDEXES             216   114    14    28   162216  52 100  50 *-*

...

SYSIBM    SYSPLAN                 79     0     6     6    41554   0 100 100 ---

SYSIBM    SYSPLANAUTH            157     0     3     3     9106   0 100 100 ---
```

```
SYSIBM    SYSPLANDEP             35     0     1     2     5985   0 100  50 --*

------------------------------------------------------------------------------


Index statistics:

F4: CLUSTERRATIO or normalized CLUSTERFACTOR > 80
F5: 100 * (KEYS * (ISIZE+8) + (CARD-KEYS) * 4) / (NLEAF * INDEXPAGESIZE) > 50
F6: (100-PCTFREE) * (INDEXPAGESIZE-96) / (ISIZE+12) ** (NLEVELS-2) * (INDEXPAGES
IZE-96) / (KEYS * (ISIZE+8) + (CARD-KEYS) * 4) < 100

CREATOR  NAME                  CARD  LEAF  LVLS ISIZE   KEYS   F4   F5  F6 REORG
------------------------------------------------------------------------------

Table: LDAPDB2.ACLPERM
LDAPDB2  ACLPERM_INDEX            2     1     1     6      2  100    -    - ---
Table: LDAPDB2.ACLPROP
LDAPDB2  ACLPROP_INDEX            2     1     1     6      2  100    -    - ---
Table: LDAPDB2.ALIASEDOBJECT
LDAPDB2  ALIASEDOBJECT            -     -     -     -      -    -    -    - ---
LDAPDB2  ALIASEDOBJECTI           -     -     -     -      -    -    -    - ---
LDAPDB2  RALIASEDOBJECT           -     -     -     -      -    -    -    - ---
Table: LDAPDB2.AUDIT
LDAPDB2  AUDITI                   1     1     1     4      1  100    -    - ---
Table: LDAPDB2.AUDITADD
LDAPDB2  AUDITADDI                1     1     1     4      1  100    -    - ---
Table: LDAPDB2.AUDITBIND
LDAPDB2  AUDITBINDI               1     1     1     4      1  100    -    - ---
Table: LDAPDB2.AUDITDELETE
LDAPDB2  AUDITDELETEI             1     1     1     4      1  100    -    - ---
Table: LDAPDB2.AUDITEXTOPEVENT
...
Table: LDAPDB2.SN
LDAPDB2  RSN                  25012   148     2    14  25012   99   90    0 ---
LDAPDB2  SN                   25012   200     3    12  25012   99   61  119 --*
LDAPDB2  SNI                  25012    84     2     4  25012   99   87    1 ---
...
Table: LDAPDB2.TITLE
LDAPDB2  TITLEI                   -     -     -     -      -    -    -    - ---
Table: LDAPDB2.UID
LDAPDB2  RUID                 25013   243     3    17  25013    0   62   79 *--
LDAPDB2  UID                  25013   273     3    17  25013  100   55   79 ---
LDAPDB2  UIDI                 25013    84     2     4  25012  100   87    1 ---
Table: LDAPDB2.UNIQUEMEMBER
LDAPDB2  RUNIQUEMEMBER        10015   224     3    47  10015    1   60   44 *--
LDAPDB2  UNIQUEMEMBER         10015   284     3    47  10015  100   47   44 -*-
LDAPDB2  UNIQUEMEMBERI        10015    14     2     4      7  100   69    8 ---

...
Table: SYSIBM.SYSFUNCTIONS
SYSIBM   IBM127                 141     1     1    13    141   65    -    - *--
SYSIBM   IBM25                  141     2     2    34    141  100   72   60 ---
SYSIBM   IBM26                  141     2     2    32    141   78   68   63 *--
SYSIBM   IBM27                  141     1     1    23     68   80    -    - *--
SYSIBM   IBM28                  141     1     1    12      2   99    -    - ---
SYSIBM   IBM29                  141     1     1     4    141  100    -    - ---
SYSIBM   IBM30                  141     3     2    59    141   78   76   38 *--
SYSIBM   IBM55                  141     2     2    34    141   99   72   60 ---
...
------------------------------------------------------------------------------


CLUSTERRATIO or normalized CLUSTERFACTOR (F4) will indicate REORG is necessary
for indexes that are not in the same sequence as the base table. When multiple
```

```
indexes are defined on a table, one or more indexes may be flagged as needing
REORG.  Specify the most important index for REORG sequencing.
```

Using the statistics generated by **reorgchk**, run **reorg** to update database table organization. See "Performing a reorg".

Keep in mind that **reorgchk** needs to be run periodically. For example, **reorgchk** might need to be run after a large number of updates have been performed. Note that LDAP tools such as **ldapadd**, **ldif2db**, and **bulkload** can potentially do large numbers of updates that require a **reorgchk**. The performance of the database should be monitored and a **reorgchk** performed when performance starts to degrade. See "Monitoring performance" on page 35 for more information.

reorgchk must be performed on all LDAP replicas because each replica uses a separate database. The LDAP replication process does not include the propagation of database optimizations.

Because LDAP caches prepared DB2 statements, you must stop and restart **ibmslapd** for DB2 changes to take effect.

Performing a reorg

After you have generated organizational information about the database using **reorgchk**, the next step in reorganization is finding the tables and indexes that need reorganizing and attempting to reorganize them. This can take a long time. The time it takes to perform the reorganization process increases as the DB2 database size increases.

In general, reorganizing a table takes more time than running statistics. Therefore, performance might be improved significantly by running statistics first.

To reorganize database table information:

1. If you have not done so already, run **reorgchk**:

   ```
   db2 reorgchk update statistics on table all > reorgchk.out
   ```

   The **reorgchk** update statistics report has two sections; the first section is the table information and the second section is the indexes. An asterisk in the last column indicates a need for reorganization.
2. To reorganize the tables with an asterisk in the last column:

   ```
   db2 reorg table table_name
   ```

   where *table_name* is the name of the table to be reorganized; for example, LDAPDB2.LDAP_ENTRY.

   Generally speaking, because most data in LDAP is accessed by index, reorganizing tables is usually not as beneficial as reorganizing indexes.
3. To reorganize the indexes with an asterisk in the last column:

   ```
   db2 reorg table table_name index index_name
   ```

   where
   - *table_name* is the name of the table; for example, LDAPDB2.LDAP_ENTRY.
   - *index_name* is the name of the index; for example, SYSIBM.SQL000414155358130.

4. Run **reorgchk** again. The output from **reorgchk** can then be used to determine whether the reorganization worked and whether it introduced other tables and indexes that need reorganizing.

Some guidelines for performing a reorganization are:

- If the number on the column that has an asterisk is close to the recommended value described in the header of each section and one reorganization attempt has already been done, you can probably skip a reorganization on that table or index.
- In the table LDAPDB2.LDAP_ENTRY there exists a LDAP_ENTRY_TRUNC index and a SYSIBM.SQL index. Preference should be given to the SYSIBM.SQL index if attempts to reorganize them seem to alternate between one or the other needing reorganization.
- Reorganize all the attributes that you want to use in searches. In most cases you will want to reorganize to the forward index, but in cases with searches beginning with '*', reorganize to the reverse index.

  For example:

```
Table: LDAPDB2.SECUUID

LDAPDB2 RSECUUID <– This is a reverse index

LDAPDB2 SECUUID <– This is a forward index

LDAPDB2 SECUUIDI <– This is an update index
```

Indexes

Indexing results in a considerable reduction in the amount of time it takes to locate requested data. For this reason, it can be very beneficial from a performance standpoint to index all attributes used in searches.

Use the following DB2 commands to verify that a particular index is defined. In the following example, the index being checked is for the attribute **principalName**:

```
db2 connect to database_name
db2 list tables for all | grep -i principalName
db2 describe indexes for table database_name.principalName
```

Where *database_name* is the name of your database.

If the second command fails or the last command does not return three entries, the index is not properly defined. The last command should return the following results:

```
IndexSchema      Index Name                 Unique Rule        Number of Columns
 -------------  --------------------        ----------         -------------
LDAPDB2         PRINCIPALNAMEI               D                  1
LDAPDB2         PRINCIPALNAME                D                  2
LDAPDB2         RPRINCIPALNAME               D                  2

             3 record (s) selected.
```

To have IBM Tivoli Directory Server create an index for an attribute the next time the server is started, do **one** of the following:

- To create an index using the Web Administration Tool:
  1. Expand **Schema management** in the navigation area, and click **Manage attributes**.
  2. Click **Edit attribute**.

3. On the **IBM extensions** tab, select the **Equality** check box under **Indexing rules**.

- To create an index from the command line, issue the following command:

```
ldapmodify -f /ldap/etc/addindex.ldif
```

The addindex.ldif file should look like this:

```
dn: cn=schema
changetype: modify
replace: attributetypes
attributetypes: ( 1.3.18.0.2.4.318 NAME ( 'principalName'  'principal' ) DESC
'A naming attribute that may be used to identify eUser object entries.' EQUALITY
 1.3.6.1.4.1.1466.109.114.2 ORDERING 2.5.13.3 SUBSTR 2.5.13.4 SYNTAX
 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
-
replace: ibmattributetypes
ibmattributetypes: ( 1.3.18.0.2.4.318 DBNAME( 'principalName'  'principalName' )
ACCESS-CLASS normal LENGTH 256 EQUALITY ORDERING SUBSTR APPROX )
```

Other DB2 configuration parameters

Performance benefits can come from setting other DB2 configuration parameters, such as APPLHEAPZ and LOGFILSIZ. The current setting of parameters can be obtained by issuing the following command:

```
db2 get database configuration for database name
```

where *database name* is the name of your database.

This command returns the settings of other DB2 configuration parameters as well.

The following command also shows the DB2 configuration parameters for the entire database instance:

```
db2 get database manager configuration
```

To set the DB2 configuration parameters use the following syntax:

```
db2 update database configuration for database name using \
parm name parm value
db2stop
db2start
```

where *database name* is the name of your database and where *parm name* is the parameter to change and *parm value* is the value it is to be assigned.

Changes to DB2 configuration parameters do not take effect until the database is restarted with **db2stop** and **db2start**.

For a list of DB2 parameters that affect performance, visit the DB2 Web site: http://www.ibm.com/software/data/db2

Note: If DB2 recognizes that a parameter is configured insufficiently, the problem is posted to the diagnostic log (db2diag.log). For example, if the DB2 buffer pools are too large, DB2 overrides the buffer pool settings and uses a minimal configuration. No notice of the change in buffer pool sizes is given except in the diagnostic log, so it is important to view the log if you are experiencing poor performance.

Database backup and restore considerations

When using the database **backup** and **restore** commands it is important to keep in mind that when you restore over an existing database, any tuning that has been done on that existing database is lost.

Chapter 4. AIX operating system tuning

This chapter discusses the following performance tuning tasks for the AIX operating system:

- Enabling large files
- Setting MALLOCMULTIHEAP
- Setting MALLOCTYPE
- Setting other environment variables
- Viewing **ibmslapd** environment variables

Enabling large files

The underlying AIX operating system files that hold the contents of a large directory can grow beyond the default size limits imposed by the AIX operating system. If the size limits are reached, the directory ceases to function correctly. The following steps make it possible for files to grow beyond default limits on an AIX operating system:

1. When you create the file systems that are expected to hold the directory's underlying files, you should create them as Large File Enabled Journaled File Systems. The file system containing the DB2 instance's home directory, and, if **bulkload** is used, the file system containing the **bulkload** temporary directory, are file systems that can be created this way.
2. Set the soft file size limit for the root, ldap, and the DB2 instance owner users to **-1**. A soft file size limit of -1 for a user specifies the maximum file size for that user as unlimited. The soft file size limit can be changed using the **smitty chuser** command. Each user must log off and log back in for the new soft file size limit to take effect. You must also restart DB2.

Setting MALLOCMULTIHEAP

The MALLOCMULTIHEAP environment variable can improve LDAP performance on symmetric multi-processor (SMP) systems. To set this variable, run the following command just before starting **ibmslapd**, in the same environment where you will start **ibmslapd**:

```
export MALLOCMULTIHEAP=1
```

The disadvantage of using MALLOCMULTIHEAP is increased memory usage.

It might take less memory, yet have less of a performance benefit, if the variable is set as follows:

```
export MALLOCMULTIHEAP=heaps: numprocs+1
```

where *numprocs* is the number of processors in the multiprocessor system.

You can find more information about MALLOCMULTIHEAP in the AIX documentation.

Setting MALLOCTYPE

Set the MALLOCTYPE environment variable as follows, according to the version of AIX you are running:

On AIX 5.1
Set MALLOCTYPE as follows:

```
export MALLOCTYPE=buckets
```

On AIX 5.2
Set MALLOCTYPE to the default. If you have already set MALLOCTYPE to another value, you can set it to the default by typing the following:

```
export MALLOCTYPE=null
```

You can find more information about MALLOCTYPE in the AIX documentation.

Setting other environment variables

You might experience better performance by setting the AIXTHREAD_SCOPE and NODISCLAIM environment as shown in the following commands. Check the AIX documentation to see if these settings might be right for your installation.

AIXTHREAD_SCOPE
To set AIXTHREAD_SCOPE, use the following command:

```
export AIXTHREAD_SCOPE=S
```

NODISCLAIM
To set NODISCLAIM, use the following command:

```
export NODISCLAIM=TRUE
```

Viewing ibmslapd environment variables (AIX operating system only)

To view the environment settings and variables for your **ibmslapd** process, run the following command:

```
ps ewww PID
```

where *PID* is the **ibmslapd** process ID.

Example output for a PID of 27594:

```
$ ps ewww  27594
   PID    TTY STAT  TIME COMMAND
    27594  pts/2 A     0:02 ibmslapd -n -f /home/gwllmz/Test/Unit/Repl/
taliesin.ibmslapd.conf _=/usr/bin/ibmslapd LANG=en_US LOGIN=root IMQCONFIGCL
/etc/IMNSearch/dbcshelp PATH=/usr/bin:/etc:/usr/sbin:/usr/local/bin:/usr/
contrib/bin:/usr/prod/bin:/usr/ucb://bin:/usr/bin/X11:/sbin:/
usr/lib/netls/conf/nodelock:.:/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/
X11:/sbin:/home/ldapdb2/sqllib/bin:/home/ldapdb2/sqllib/adm:/home/ldapdb2/
sqllib/misc ID=root LC__FASTMSG=true IMQCONFIGSRV=/
etc/IMNSearch CGI_DIRECTORY=/var/docsearch/cgi-bin CLASSPATH=/home/ldapdb2/
sqllib/java/db2java.zip:/home/ldapdb2/sqllib/java/db2jcc.jar:/home/ldapdb2/
sqllib/java/sqlj.zip:/home/ldapdb2/sqllib/function:
. LOGNAME=root MAIL=/usr/spool/mail/root LOCPATH=/usr/lib/nls/
loc HNAME=taliesin PS1=[$ID@$HNAME: $PWD ]?==>  OS=AIX VWSPATH=/home/
ldapdb2/sqllib DOCUMENT_SERVER_MACHINE_NAME=localhost
 USER=root AUTHSTATE=compat _EUC_SVC_DBG_LOGGING=1 DCE_USE_WCHAR_NAMES=
1 _EUC_SVC_DBG_FILENAME=/tmp/gss.out SHELL=/bin/ksh ODMDIR=/etc/objrepos JAVA_HOME=/
usr/jdk_base DOCUMENT_SERVER_PORT=49213
 _EUC_SVC_DBG=*.8 HOME=/ DB2INSTANCE=ldapdb2 VWS_TEMPLATES=/home/
ldapdb2/sqllib/templates LD_LIBRARY_PATH=:/home/ldapdb2/sqllib/lib TERM=vt220 MAILMSG=
[YOU HAVE NEW MAIL] PWD=/home/gwllmz/Defects/77260/aus51ldap.20021106a DOCUMENT_DIRECTORY=/
```

```
usr/docsearch/html TZ=CST6CDT
TRY_PE_SITE=1 VWS_LOGGING=/home/ldapdb2/sqllib/logging A__z=! LOGNAME LIBPATH=/
usr/lib:/usr/ldap/java/bin:/usr/ldap/java/bin/classic:/home/ldapdb2/sqllib/
lib NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/%L/%N.cat KRB5_KTNAME=/home/
gwllmz/Test/Unit/Repl/taliesin.keytab
 ODBCCONN=15
```

Chapter 5. Hardware tuning

This chapter contains some suggestions for improving disk drive performance.

Disk speed improvements

With millions of entries in LDAP server, it can become impossible to cache all of them in memory. Even if a smaller directory size is cacheable, update operations must go to disk. The speed of disk operations is important. Here are some considerations for helping to improve disk drive performance:

- Use fast disk drives
- Use a hardware write cache
- Spread data across multiple disk drives
- Spread the disk drives across multiple I/O controllers
- Put log files and data on separate physical disk drives

Chapter 6. IBM Tivoli Directory Server features

The sections in this chapter briefly describe the following additional performance-related IBM Tivoli Directory Server features.

- Bulk loading (**bulkload**)
- Replication
- Monitoring Performance
- When to configure the LDAP change log

Bulk loading (bulkload)

The **bulkload** utility loads directory data to the LDAP database using an LDAP Data Interchange Format (LDIF) file. **bulkload** usually is significantly faster than **ldif2db** and **ldapadd** when loading approximately 100,000 to a million entries. Read the **bulkload** documentation in the *IBM Directory Server Version 5.1 Administration Guide* for information about using **bulkload**.

Replication

IBM Tivoli Directory Server supports replication. Through replication, multiple, synchronized copies of directory data are maintained on multiple directory servers. Using replication can improve performance related to search throughput because the workload can be divided across several servers. This allows for an enterprise's LDAP search activity to be distributed among several servers.

In the configuration of replication, you can specify that updates be replicated either immediately or at scheduled times. If your application environment does not depend on the updates occurring immediately, it might be beneficial to schedule replication for off-peak hours, or on some more frequent, periodic basis. This will cause the updates to be batched together, reducing the cost of cache invalidation that results from updates on the replica servers.

Replication also adds to the workload on the master server where the updates are first applied. In addition to updating its copy of the directory data, the master server must send the changes to all replica servers. Careful scheduling of replication to avoid peak activity times will help minimize the impact to throughput on the master server.

Monitoring performance

The **ldapsearch** command can be used to monitor performance, as shown in the following sections.

ldapsearch with "cn=monitor"

The following **ldapsearch** command uses "cn=monitor".

```
ldapsearch -h ldop_host -s base -b cn=monitor objectclass=*
```

where *ldap_host* is the name of the LDAP host.

The monitor search returns some of the following attributes of the server:

cn=monitor

version=IBM Tivoli Directory, Version 5.2

total connections
The total number of connections since the server was started.

current connections
The number of active connections.

maxconnections
The maximum number of active connections allowed.

writewaiters
The number of threads sending data back to the client.

readwaiters
The number of threads reading data from the client.

opsinitiated
The number of initiated requests since the server was started.

livethreads
The number of worker threads being used by the server.

opscompleted
The number of completed requests since the server was started.

entriessent
The number of entries sent by the server since the server was started.

searchesrequested
The number of initiated searches since the server was started.

searchescompleted
The number of completed searches since the server was started.

filter_cache_size
The maximum number of filters allowed in the cache.

filter_cache_current
The number of filters currently in the cache.

filter_cache_hit
The number of filters retrieved from the cache rather than being resolved in DB2.

filter_cache_miss
The number of filters that were not found in the cache that then needed to be resolved by DB2.

filter_cache_bypass_limit
Search filters that return more entries than this limit are not cached.

entry_cache_size
The maximum number of entries allowed in the cache.

entry_cache_current
The number of entries currently in the cache.

entry_cache_hit
The number of entries that were retrieved from the cache.

entry_cache_miss
The number of entries that were not found in the cache that then needed to be retrieved from DB2.

acl_cache
A Boolean value indicating that the ACL cache is active (TRUE) or inactive (FALSE).

acl_cache_size
The maximum number of entries in the ACL cache.

currenttime
The current time on the server. The current time is in the format:

```
year month day hour:minutes:seconds GMT
```

Note: If expressed in local time the format is

```
day month date hour:minutes:seconds timezone year
```

starttime
The time the server was started. The start time is in the format:

```
year month day hour:minutes:seconds GMT
```

Note: If expressed in local time the format is

```
day month date hour:minutes:seconds timezone year
```

en_currentregs
The current number of client registrations for event notification.

en_notificationssent
The total number of event notifications sent to clients since the server was started.

The following attributes are for operation counts:

bindsrequested
The number of bind operations requested since the server was started.

bindscompleted
The number of bind operations completed since the server was started.

unbindsrequested
The number of unbind operations requested since the server was started.

unbindscompleted
The number of unbind operations completed since the server was started.

addsrequested
The number of add operations requested since the server was started.

addscompleted
The number of add operations completed since the server was started.

deletesrequested
The number of delete operations requested since the server was started.

deletescompleted
The number of delete operations completed since the server was started.

modrdnsrequested
The number of modify RDN operations requested since the server was started.

modrdnscompleted
The number of modify RDN operations completed since the server was started.

modifiesrequested
The number of modify operations requested since the server was started.

modifiescompleted
The number of modify operations completed since the server was started.

comparesrequested
The number of compare operations requested since the server was started.

comparescompleted
The number of compare operations completed since the server was started.

abandonsrequested
The number of abandon operations requested since the server was started.

abandonscompleted
The number of abandon operations completed since the server was started.

extopsrequested
The number of extended operations requested since the server was started.

extopscompleted
The number of extended operations completed since the server was started.

unknownopsrequested
The number of unknown operations requested since the server was started.

unknownopscompleted
The number of unknown operations completed since the server was started.

The following attributes are for server logging counts:

slapderrorlog_messages
The number of server error messages recorded since the server was started or since a reset was performed.

slapdclierrors_messages
The number of DB2 error messages recorded since the server was started or since a reset was performed.

auditlog_messages
The number of audit messages recorded since the server was started or since a reset was performed.

auditlog_failedop_messages
The number of failed operation messages recorded since the server was started or since a reset was performed.

The following attributes are for connection type counts:

total_ssl_connections
The total number of SSL connections since the server was started.

total_tls_connections
The total number of TLS connections since the server was started.

The following attributes are for tracing:

trace_enabled
The current trace value for the server. TRUE, if collecting trace data, FALSE, if not collecting trace data.

trace_message_level
The current ldap_debug value for the server. The value is in hexadecimal form, for example:

```
0x0=0
0xffff=65535
```

trace_message_log
The current LDAP_DEBUG_FILE environment variable setting for the server.

The following attributes are for denial of service prevention:

available_workers
The number of worker threads available for work.

current_workqueue_size
The current depth of the work queue.

largest_workqueue_size
The largest size that the work queue has ever reached.

idle_connections_closed
The number of idle connections closed by the Automatic Connection Cleaner.

auto_connection_cleaner_run
The number of times that the Automatic Connection Cleaner has run.

emergency_thread_running
The indicator of whether the emergency thread is running.

totaltimes_emergency_thread_run
The number of times the emergency thread has been activated.

lasttime_emergency_thread_run
The last time the emergency thread was activated.

The following attribute has been added for alias dereference processing:

bypass_deref_aliases
The server runtime value that indicates if alias processing can be bypassed. It displays TRUE if no alias object exists in the directory, and FALSE if at least one alias object exists in the directory.

The following attributes are for the attribute cache:

cached_attribute_total_size
The amount of memory used by the directory attribute cache, in kilobytes. This number includes additional memory used to manage the cache that is not charged to the individual attribute caches. Consequently, this total is larger than the sum of the memory used by all the individual attribute caches.

cached_attribute_configured_size
The maximum amount of memory, in kilobytes, assigned to the directory attribute cache.

cached_attribute_hit
The number of times the attribute has been used in a filter that could be processed by the attribute cache. The value is reported as follows:

```
cached_attribute_hit=attrname:#####
```

cached_attribute_size

The amount of memory used for this attribute in the attribute cache. This value is reported in kilobytes as follows:

```
cached_attribute_size=attrname:######
```

cached_attribute_candidate_hit

A list of up to ten most frequently used noncached attributes that have been used in a filter that could have been processed by the directory attribute cache if all of the attributes used in the filter had been cached. The value is reported as follows:

```
cached_attribute_candidate_hit=attrname:#####
```

You can use this list to help you decide which attributes you want to cache. Typically, you want to put a limited number of attributes into the attribute cache because of memory constraints.

Examples

The following sections show examples of using values returned by the **ldapsearch** command with "cn=monitor" to calculate the throughput of the server and the number of add operations completed on the server in a certain timeframe.

Throughput example: The following example shows how to calculate the throughput of the server by monitoring the server statistic called **opscompleted**, which is the number of operations completed since the LDAP server started.

Suppose the values for the **opscompleted** attribute obtained by issuing two **ldapsearch** commands to monitor the performance statistics, one at time **t1** and the other at a later time **t2**, were opscompleted (**t1**) and opscompleted(**t2**.) The average throughput at the server during the interval between **t1** and **t2** can be calculated as:

```
(opscompleted(t2) - opscompleted(t1) - 3)/(t2 -t1)
```

(3 is subtracted to account for the number of operations performed by the **ldapsearch** command itself.)

Workload example: The monitor attributes can be used to characterize the workload, similar to the throughput example but split out by type of operation. For example, you can calculate the number of add operations that were completed in a certain amount of time.

Suppose the values for the **addscompleted** attribute obtained by issuing two **ldapsearch** commands to monitor the performance statistics, one at time **t1** and the other at a later time **t2**, were addscompleted (**t1**) and addscompleted(**t2**.) The number of add operations completed on the server during the interval between **t1** and **t2** can be calculated as:

```
(addscompleted(t2) - addscompleted(t1) - 3)/(t2 -t1)
```

(3 is subtracted to account for the number of operations performed by the **ldapsearch** command itself.)

Similar calculations can be done for other operations, such as **searchescompleted**, **bindscompleted**, **deletescompleted**, and **modifiescompleted**.

ldapsearch with "cn=workers,cn=monitor"

You can run a search using "cn=workers,cn=monitor" to get information about what worker threads are doing and when they started doing it.

```
ldapsearch -D <adminDN> -w <adminpw> -b cn=workers,cn=monitor -s base objectclass=*
```

This information is most useful when a server is performing poorly or not functioning as expected. It should be used only when needed to give insight into what the server is currently doing or not doing.

The "cn=workers, cn=monitor" search returns detailed activity information only if auditing is turned on. If auditing is not on, "cn=workers, cn=monitor" returns only thread information for each of the workers.

Attention: The "cn=workers,cn=monitor" search suspends all server activity until it is completed. For this reason, a warning should be issued from any application before issuing this feature. The response time for this command will increase as the number of server connections and active workers increase.

For more information, see the *IBM Tivoli Directory Server Version 5.2 Administration Guide*.

ldapsearch with "cn=connections,cn=monitor"

You can run a search using "cn=connections,cn=monitor" to get information about server connections:

```
ldapsearch -D<adminDN> -w <adminPW> -h <servername> -p <portnumber>
         -b cn=connections,cn=monitor -s base objectclass=*
```

This command returns information in the following format:

```
cn=connections,cn=monitor
connection=1632 : 9.41.21.31 : 2002-10-05 19:18:21 GMT  : 1 : 1 : CN=ADMIN : :
connection=1487 : 127.0.0.1 : 2002-10-05 19:17:01 GMT  : 1 : 1 : CN=ADMIN : :
```

Note: If appropriate, an SSL or a TLS indicator is added on each connection.

For more information, see the *IBM Tivoli Directory Server Version 5.2 Administration Guide*.

ldapsearch with "cn=changelog,cn=monitor"

You can run a search using "cn=changelog,cn=monitor" to obtain information about the changelog attribute cache. (See "When to configure the LDAP change log" on page 42 for information about the change log.) The command returns the following information:

cached_attribute_total_size
: The amount of memory used by the changelog attribute cache, in kilobytes. This number includes additional memory used to manage the cache that is not charged to the individual attribute caches. Consequently, this total is larger than the sum of the memory used by all the individual attribute caches.

cached_attribute_configured_size
: The maximum amount of memory, in kilobytes, assigned to the changelog attribute cache.

cached_attribute_hit
: The number of times the attribute has been used in a filter that could be processed by the changelog attribute cache. The value is reported as follows:

```
cached_attribute_hit=attrname:#####
```

cached_attribute_size

The amount of memory used for this attribute in the changelog attribute cache. This value is reported in kilobytes as follows:

```
cached_attribute_size=attrname:######
```

cached_attribute_candidate_hit

A list of up to ten most frequently used noncached attributes that have been used in a filter that could have been processed by the changelog attribute cache if all of the attributes used in the filter had been cached. The value is reported as follows:

```
cached_attribute_candidate_hit=attrname:#####
```

You can use this list to help you decide which attributes you want to cache. Typically, you want to put a limited number of attributes into the attribute cache because of memory constraints.

When to configure the LDAP change log

IBM Tivoli Directory Server 5.2 has a function called **change log** that results in a significantly slower LDAP update performance. The **change log** function should be configured only if needed.

The **change log** function causes all updates to LDAP to be recorded in a separate change log DB2 database (that is, a different database from the one used to hold the LDAP server Directory Information Tree). The change log database can be used by other applications to query and track LDAP updates. The change log function is disabled by default.

One way to check for existence of the **change log** function is to look for the suffix CN=CHANGELOG. If it exists, the **change log** function is enabled.

Chapter 7. Troubleshooting

If you are experiencing problems with the performance of your directory server, refer to this section for possible fixes and workarounds.

Identifying performance problem areas

This section contains some methods for identifying areas that might be affecting the performance of your directory server.

Audit log

The audit log shows what searches are being performed and the parameters used in each search. The audit log also shows when a client binds and unbinds from the directory. Observing these measurements allows you to identify LDAP operations that take a long time to complete.

ibmslapd trace

An ibmslapd trace provides a list of the SQL commands issued to the DB2 database. These commands can help you identify operations that are taking a long time to complete. This information can in turn lead you to missing indexes, or unusual directory topology. To turn the ibmslapd trace on, run the following commands:

1. `ldtrc on`
2. `ibmslapd –h 4096`

After you have turned the trace on, run the commands that you think might be giving you trouble.

Running a trace on several operations can slow performance, so remember to turn the trace off when you are finished using it:

```
ldtrc off
```

Adding memory after installation on Solaris systems

Memory added to a computer after the installation of a Solaris operating system does not automatically improve performance. To take advantage of added memory, you must:

1. Update the shared memory (shmem) value in the /etc/system file:

   ```
   set shmsys:shminfo_shmmax = physical_memory
   ```

   Where *physical_memory* is the size on of the physical memory on the computer in bytes.

   You must restart the computer for the new settings to take effect.

2. From the command line, set the ulimit values for increasing process memory and file size to unlimited:

   ```
   ulimit -d unlimited
   ulimit -v unlimited
   ulimit -f unlimited
   ```

Setting the SLAPD_OCHANDLERS environment variable on Windows

On Windows, if you have clients that are generating many connections to the server and the connections are being refused, set the SLAPD_OCHANDLERS environment variable to 5 before starting the server.

DB2 rollbacks and isolation levels

If you are experiencing rollback activities in DB2, check the isolation level. Rollbacks occur when one application process has a row locked while another application process tries to access that same row. Because the default isolation level, repeatable read, can result in more rows being locked than are actually required for the current read request, a more relaxed isolation level is normally recommended for LDAP applications.

For example, the read stability isolation level allows other applications to insert or update data in rows that have been read. If a second read is issued for that range of rows, the new data is reflected in the result set. Keep in mind, however, that the second read can return data that is different from the first read. If an application depends upon the same data being returned on multiple reads, the isolation level should be set to repeatable read.

To set the DB2 isolation level, type the following at a command prompt:

```
db2set <isolation_level>=YES
```

where *isolation_level* is the isolation level you want to apply, such as DB2_RR_TO_RS.

Note: All applications using the current database instance are affected by this setting.

Default value of LOGFILSIZ needs to be increased

If you are adding a very large group (more than 50,000 members) to your 5.2 directory, and you have migrated your database from a previous release such as Version 4.1, modify the LOGFILSIZ parameter of your DB2 database to be at least 2000. On migrated databases, this value might currently be set to 750 or 1000.

You can verify this value by issuing the following commands. For this example the names of the user, instance, and database are **ldapdb2**.

For UNIX platforms:

```
su -ldapdb2
db2start
db2 get database config for ldapdb2 | grep LOGFILSIZ
```

To increase this value, issue the following command:

```
db2 update database config for ldapdb2 using LOGFILSIZ 2000
```

For Windows platforms:

```
db2cmd
set DB2INSTANCE=ldapdb2
db2 get database config for ldapdb2 ><outputfile>
```

Find the value for LOGFILSIZ in the output file. To increase this value, issue the following command:

```
db2 update database config for ldapdb2 using LOGFILSIZ 2000
```

Note: This value is already set correctly if you created or configured your database with the IBM Tivoli Directory Server Version 5.2 Configuration Tool.

Appendix A. DirectoryMark 1.2

DirectoryMark 1.2 is an industry-standard benchmark provided by Mindcraft Inc. as a performance test for LDAP implementations.

http://www.mindcraft.com/directorymark/

DirectoryMark 1.2 consists of a loading phase and two scenarios, Messaging and Address Lookup. Both Messaging and Address Lookup consist entirely of searches; there are no updates.

Each scenario consists of two phases, a warmup phase and a run phase. During the warmup phase, the searches primarily request entries that are not in the LDAP caches; most of these requests require interaction with the DB2 backing store. For all the measurements reported in this document, warmup consisted of running all queries at least once; consequently, during the run phase all entries requested are potentially already in LDAP caches in memory if the caches are large enough to hold all of them. Thus the warmup phase and the run phase comprise two distinctly different workloads.

During the run phase of Address Lookup, a number of client threads issue search requests to the IBM Tivoli Directory Server from predetermined scripts. The scripts include a number of different kinds of searches, including wildcard and other searches that return multiple entries per request. The client threads run through their scripts continuously for three minutes. Throughput is measured on the server for each three-minute interval, and then each client starts over at the beginning of its script. Each three-minute interval is referred to as a run. The server is not restarted between runs.

Appendix B. Platform configurations

The examples in this guide use the following platform configurations:

- Clients
  - Two 866 MHz Intel PIII with 512 MB RAM; IBM 10/100 EtherJet™ PCI Adapter (1 processor active)
  - Two 4 CPU 1.8 GHz Intel PIV with 1GB RAM; Intel PRO/100 VE Desktop Connection (when 4 processors active)
  - Windows 2000 Professional with SP2®
- Server
  - 4-Way 600 MHz, pSeries™ eServer™ Model 660-6H1 with 1 or 4 processor active, 4 GB RAM.
  - IBM 10/100 Ethernet Adapter.
  - AIX 5.1
  - IBM Directory Server Version 4.1 GA
  - AIXTHREAD_SCOPE=S
  - MALLOCTYPE=3.1 (1way) or MALLOCMULTIHEAP=1 (4 way).
  - NODISCLAIM=true (1way).
  - ACLCACHESIZE=150000.
  - RDBM_CACHE_SIZE=460000 except where noted.
  - RDBM_FCACHE_SIZE=75000 except where noted.
  - RDBM_CACHE_BYPASS_LIMIT=100 except where noted.
  - Necessary Indexes created (for attribute seeAlso).
  - No ACLs were set. By default, anyone can search and compare. The directory administrator can update.
- DB2 v 7.2
  - maxlocks 100 sortheap 2500 dbheap 5000 ibmdefaultbp 236000 (4K pages) ldapbp 9800 (32K pages)
  - logfilsiz 2048, logprimary 6
- Miscellaneous
  - Caches were warmed up by running all scripts once.
  - Measurements were taken using 50 client threads except where noted.
  - DB2 log files are on the same disk as the containers.
  - The DirectoryMark version is 1.2.

Appendix C. Notices

This information was developed for products and services offered in the U.S.A. IBM might not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department MU5A46
11301 Burnet Road
Austin, TX 78758
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

AIX
Database 2
DB2
DB2 Universal Database
eServer
EtherJet
IBM
pSeries
SecureWay
SP2
Tivoli

Intel, Intel Inside (logos), MMX™ and Pentium® are trademarks of Intel Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Microsoft®, MS-DOS, Windows, and Windows NT® are registered trademarks of Microsoft Corporation

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

Index

T

IBM®

Printed in U.S.A.

SC32-1342-00

